Universidad Rey Juan Carlos

Grado en Ingeniería en Telemática

Escuela Técnica Superior de Ingeniería Telecomunicación

Curso académico 2017/2018

Trabajo Fin de Grado

Robot submarino OpenROV y ROS

Autor: Saúl Ibáñez Cerro

Tutor: Francisco Martín Rico

A

Agradecimientos

Primero, quiero agradecer a mis padres todo el esfuerzo que han realizado por mí, por haberme apoyado durante toda mi vida académica, muchas gracias por estar siempre ahí, por quererme y por animarme cuando más lo necesitaba. También quiero agradecer al "enano" de mi corazón, mi hermano, con el que siempre he podido hablar de mis quebraderos de cabeza. Por supuesto, también merece una mención especial mi pareja, Paula, no sé si hubiera podido con la presión y desánimos si no fuera por ella. Me ha ayudado más que nadie, le debo muchísimo. Otro apoyo muy grande que he tenido han sido los padres de mi novia, que han estado siempre ahí cerca ayudándome. A mis abuelos por apoyarme y a todos los amigos y compañeros que me han ayudado durante toda esta etapa. Para finalizar quiero mostrar mi gratitud a un profesor que me ha marcado, por la forma de enseñar y mostrarnos su asignatura, el profesor de robótica, Francisco Martín, al que también agradezco darme la oportunidad de hacer un TFG distinto.

Muchas gracias a todos. Todos los éxitos que consiga en la vida, serán en parte gracias a vosotros.

Índice general

1. Introducción **1**

1.1. Robótica móvil 1

1.2. Robots Submarinos 3

1.3. ROS 5

1.4. Estructura de la memoria 6

2. Objetivos y Metodología **7**

2.1. Descripción del problema y requisitos 7

2.2. Objetivo del proyecto 7

2.3. Metodología de desarrollo 8

2.4. Plan de trabajo 9

3. Plataforma OpenROV **10**

3.1. Robot Submarino OpenROV 10

3.2. Estructura y Montaje 11

3.2.1. Estructura de OpenROV 11

3.2.2. Tapas del chásis 12

3.2.3. Electrónica 13

3.2.4. Enrutamiento 15

3.2.5. Final 16

3.3. Conectividad OpenROV 16

3.3.1. Diseño de Conectividad 16

3.3.2. Puesta a punto de los elementos 20

3.3.3. Web/Cockpit 23

3.3.4. Mantenimiento 27

3.3.5. Problemas 28

4. Integración de ROS en OpenROV **31**

4.1. Configuración del ROV 31

4.2. Configuración de ROS 35
4.3. Conexión de ROS en OpenROV 37

5. Pruebas de Campo **41**

6. Conclusiones **45**

Índice de figuras

1.1. Robots submarinos 4
1.2. Roscore 6

2.1. Modelo en cascada. 8

3.1. OpenROV 11
3.2. Piezas de los chásis. 11
3.3. Chasis principal. 12
3.4. Soporte de la cámara y electrónica 12
3.5. Tapa inferior 13
3.6. Raspberry Pi + PLC + BeagleBone 14
3.7. Electrónica de la cámara 14
3.8. Soldadura de cables 15
3.9. PLC externo 17
3.10. BeagleBone configurada 18
3.11. Conexión Windows 8 18
3.12. Actualización firmware 20
3.13. Foco cámara 20
3.14. Láser 21
3.15. Motores 21
3.16. Diagnostics 22
3.17. Tubo 22
3.18. Tapa superior 23
3.19. Cockpit (1) 23
3.20. Cockpit (2) 24
3.21. Cockpit (3) 25
3.22. Ajustes 25
3.23. Diagnóstico 26
3.24. Servicios 27

3.25. DB-25 29

4.1. Diagrama de la conexión ROS y OpenROV 32
4.2. SSH 32
4.3. Dependencias Roslibjs 33
4.4. Canvas 34
4.5. Roscore 38
4.6. Launch 38
4.7. Conexión en ROV 39
4.8. rostopic list 39
4.9. rostopic info (1) 40
4.10. rostopic info (2) 40

5.1. Prueba de motores 41
5.2. Prueba de funcionamiento en la bañera 42
5.3. Prueba de funcionamiento en el lago 43
5.4. Prueba de navegación en el lago 43
5.5. Prueba de emersión del lago 44
5.6. Prueba de visualización del lago en la web 44

Capítulo 1

Introducción

En este capítulo se explicará el concepto de robótica móvil [2]. También se comentarán los robots submarinos, ya que en este proyecto, trataremos sobre uno de ellos, OpenROV. Hablaremos sobre ROS (Robot Operating System), que tiene un conjunto de herramientas y atajos que facilitan la comunicación y operabilidad de los robots, ya que en uno de los capítulos del proyecto, se expondrá la integración de este framework en el ROV. Para finalizar se describirá la estructura de la memoria y de los apartados que conlleva.

1.1. Robótica móvil

Generalmente las aplicaciones robóticas se centraban en los sectores manufactureros más desarrollados para la producción masiva: industria del automóvil, transformaciones metálicas, etc.

A principios de los años sesenta se introducen en la industria los robots manipuladores como un elemento más del proceso productivo. Los trabajos desarrollados por los robots manipuladores consistían frecuentemente en tareas repetitivas.

Un robot móvil puede moverse sobre entornos no estructurados, de los cuales, no se posee conocimiento. Esto lo realiza mediante la interpretación de los datos obtenidos a través de los sensores y del estado actual del vehículo.

En general se considera que existen tres clases de robots:

- **Industriales**, son los de mayor difusión en tareas de alcance económico, formados por una estructura mecánica articulada, que se mueve adoptando distintas configuraciones por las órdenes recibidas de un equipo de control, generalmente, un microprocesador.

- **Médicos**, se diseñan prótesis inteligentes para la rehabilitación de disminuidos físicos. Se procura que estas prótesis tengan la estética correspondiente a la extremidad perdida.

- **Móviles**, tienen una plataforma mecánica dotada de un sistema de locomoción capaz de navegar a través de un determinado ambiente de trabajo.

  Este tipo de robots estan dotados de un sistema de locomoción, el cual, es capaz de navegar en distintos entornos de trabajo. La locomoción puede estar diseñada con patas (equilibrio estático o dinámico) o con ruedas, que al final, son más eficientes que las piernas y son estáticamente estables.

  El mayor inconveniente de la locomoción es el desplazamiento a un lugar concreto, planificar las trayectrias y seguir la una determinada trayectoria puede resultar imposible para algunos robots.

  Un robot móvil autónomo se caracteriza por una conexión inteligente entre las operaciones de percepción y actuación.

  - **Percepción**, es la capacidad del robot móvil para gestionar la información obtenida a través de los sensores (u otros medios) con el objetivo de lograr una navegación global o local.

    - Navegación global es aquella que se apoya en mapas (SLAM). Existen varios tipos, entre los que encontramos:

      1. Planificación de los caminos (planes clásicos (secuencia de subobjetivos) y planes con recursos (por donde se le recomienda ir)).
      2. Navegación basada en mapas.
      3. Listas de caminos, grafos de objetos o balizas.
      4. Grafos de visibilidad.
      5. Diagramas de Voroni.
      6. Descomposición en celdas.
      7. Planificación como descenso del gradiente.

    - Navegación local es aquella que puede navegar si un mapa, y las tecnologías más usadas en ellas son:

      1. Método de velocidad y curvatura (CVM), el cual, trabaja añadiendo restricciones al espacio de velocidad y escogiendo un punto del

espacio que satisfaga todas las restricciones y maximice una función objetivo.

2. Método de carriles y velocidad (LVM), esta tecnología, divide el entorno en carriles para decidir cual es el mejor camino a seguir.
3. Método de fuerzas virturales (VFF), tanto los obstaculos como el destino ofrecen un tipo de fuerza, repulsiva y atractiva respectivamente. Al final, el movimiento del robot está gobernado por el sumatorio vectorial de dichas fuerzas.

- **Actuación**, el robot móvil tiene que de decidir que acción es la idónea en cada momento, dependiendo del entorno y de su estado.

1.2. Robots Submarinos

En este apartado, se hará una breve introducción sobre los robots submarinos (en este trabajo nos centraremos en uno de ellos, OpenROV 2.8[1]) y donde pueden resultar de mayor utilidad.

El mundo de los robots submarinos ROV (*Remote Operated Vehicle* o vehículo de operación remota) sigue creciendo año tras año. El 71 % de la superficie de la Tierra está cubierta de océanos y sin embargo, solo se ha mapeado el 5 % del suelo oceánico.

Durante los últimos años, el uso de robots submarinos ha aumentado rápidamente, ya que este vehículo puede operar en áreas más profundas y más peligrosas, a los cuales los humanos no podemos llegar. Este tipo de robot se pueden utilizar para el área de la pesca, control submarino de contaminación, manipulsación y limpieza del océano, así como de sitios nucleares.

Se trata de un área de diversas aplicaciones en el mundo real, ya que permite y facilita la exploración, estudios científicos de zonas inexploradas, la búsqueda y el rescate de navíos.

Podriamos diferenciar las diferentes áreas potenciales en las cuales tienen una aplicación como:

- **Ciencia**

  Generación de mapas del suelo marino.

  Rápida respuesta a eventos eceanográficos y geotérmicos.

  Muestreo geológico

[1] https://www.openrov.com/products/openrov28/

- **Militar**

  Búsqueda y eliminación de minas submarinas.

  Reconocimiento de rutas, áreas y zonas.

  Vigilancia.

  Búsqueda y rescate.

  Adquisición de objetos.

  Seguridad en las rutas.

  Soporte en el desarrollo de la situación.

  Soporte en la preparación del campo de batalla.

- **Minería oceánica e industria del petróleo**

  Estudio del océano y evaluación de recursos.

  Construcción y mantenimiento de estructuras submarinas.

- **Otras**

  Inspección del casco en los barcos.

  Inspección de plantas nucleares.

  Instalación e inspección de cables.

  Rutas turísticas bajo el mar.

  Etc.

(a) Robot Deep Trekker (b) Robot OpenROV

Figura 1.1: Robots submarinos

1.3. ROS

Sistema Operativo de Robots (en inglés *Robot Operating System, ROS* [3]) es un framework para el desarrollo de software para robots.

Implementaremos ROS en el robot submarino ya que es el framework por excelencia para el desarrollo de las aplicaciones robóticas. Contiene algoritmos implementados, además de proporcionar una arquitectura distribuida. No hace falta que los nodos de ROS interactuen en el mismo sistema ni que deban ser de la misma arquitectura, por eso es una gran ventaja utilizarlo en la tecnología de OpenROV.

ROS provee los servicios estándar de un sistema operativo tales como abstracción del hardware, control de dispositivos de bajo nivel, implementación de funcionalidad de uso común, paso de mensajes entre procesos y mantenimiento de paquetes. Está basado en una arquitectura de grafos donde el procesamiento toma lugar en los nodos que pueden recibir, mandar y multiplexar mensajes de sensores, control, estados, planificaciones y actuadores, entre otros.

ROS tiene dos partes básicas: la parte del sistema operativo, ROS, y una suite de paquetes aportados por la contribución de usuarios que implementan la funcionalidad, como localización y mapeo simultáneo, planificación, percepción, simulación, etc.

ROS es OpenSource bajo términos de licencia BSD. Esta licencia permite libertad para uso comercial e investigador.

ROS utiliza de topics como modelo de comunicación entre procesos, a los que se les nombra como nodos. A continuación se detallan los conceptos principales de ROS:

1. **Master**. Proporciona el registro de nombre. Sirve para encontrar los nodos, intercambiar mensajes e invocar servicios. Para ejecutar el ROS Master, utilizaremos el comando:

   1.1: roscore

   ```
   $ roscore
   ```

   Y se visualizará lo siguiente:

   A partir de ahora, el sistema ya sabe donde encontrar los comandos para ejecutar ROS.

2. **Nodos (Node)**. Son procesos de comunicación. Por ejemplo, un nodo controla la cámara, otro nodo controla los motores, etc. En general, los nodos son ejecutables que se comunican con otros procesos, a los que denominamos topics.

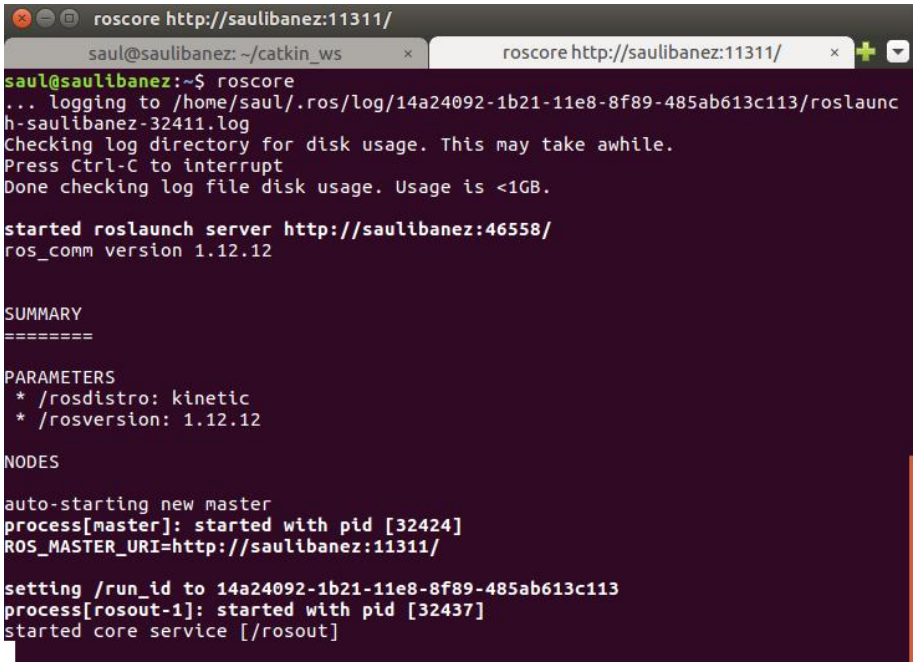


Figura 1.2: Roscore

3. **Mensajes**. Son estructuras de datos compuestos (integer, float, string, arrays, etc).

4. **Topics**. Es un canal de comunicación para transmitir los datos, el cual uiliza el modo de suscriptor y publicador. Puede existir varios publicadores para un único tópico, al igual que un nodo puede publicar y/o suscribirse a múltiples topics. Se transimite utilizando la tecnología de TCP/IP o UDP.

1.4. Estructura de la memoria

En este documento se describen los aspectos más relevantes del desarrollo y montaje del robot. La memoria esta dividida en 6 capítulos. El primer capítulo, Introducción, se ha realizado una presentación de los robots móviles. En el segundo capítulo se define el problema y se establecen los objetivos. El montaje del robot, el entorno y las herramientas utilizadas se especifica en el tercer capítulo. En el siguiente capítulo, el cuarto, se mostrará la integración del OpenROV con ROS. En el quinto capítulo se expondrán una serie de videos comprobando el funcionamiento del OpenROV, y en el último, el sexto, expondré las conclusiones del trabajo.

Capítulo 2

Objetivos y Metodología

Tras haber presentado el contexto en el que se desarrollará este proyecto, en este capítulo se fijarán los objetivos y se expondrán los requisitos para llegar a la solución.

2.1. Descripción del problema y requisitos

El fondo oceánico es uno de los grandes misterios de la actualidad. La mayor parte del suelo oceánico no ha sido explorado, y eso supone un gran problema. Con este robot, podremos rozar un poco este mundo submarino tan fascinante, e intentar descubrir algunas de las magníficas maravillas que esconde.

El robot contiene una cámara de alta resolución que permite visualizar el suelo oceánico. Cuando la visión de la cámara sea baja o nula, precisaremos de las luces y los láseres incorporados, con ellos, podremos visualizar mejor el entorno y saber la distancia a la que se encuentren los objetos, plantas o animales.

Los requisitos indispensables para realizar el proyecto son:

1. **Impermeabilidad** durante todo el proceso de sumersión
2. **Conectividad.** El robot puede alcanzar una distancia de 100m sin que pierda señal.
3. **Duración de las baterías.** El robot debe permanecer activo durante 6 horas.
4. **Web Cockpit.** Control del funcionamiento del robot.

2.2. Objetivo del proyecto

El objetivo del proyecto es el montaje, configuración del software de ROV y ROS, y unas pruebas de campo del robot submarino OpenROV.
Se montará de cero el robot submarino, utilizando los materiales dados por la

comunidad de OpenROV. A la placa de OpenROV se le instalará un PLC para la comunicación del robot con el portátil y otra placa BeagleBone, que llevará implantado el software necesario para la comunicación y funcionamiento con el robot.
Además, se implantará una biblioteca de ROS dentro de OpenROV para que también se pueda comunicar a través del framework de robótica más famoso. Con esta biblioteca instalada y ejecutando el ROS Master en el portátil, se podrá realizar una comunicación.

2.3. Metodología de desarrollo

En el desarrollo del sistema descrito, el modelo de ciclo de vida utilizado ha sido el modelo Modelo en cascada [4].

Es un proceso secuencial, fácil de desarrollo en el que los pasos de desarrollo son vistos hacia abajo (como en una cascada de agua) a través de las fases de análisis de las necesidades, el diseño, implantación, pruebas (validación), la integración, y mantenimiento.

Los principios básicos del modelo de cascada son los siguientes:

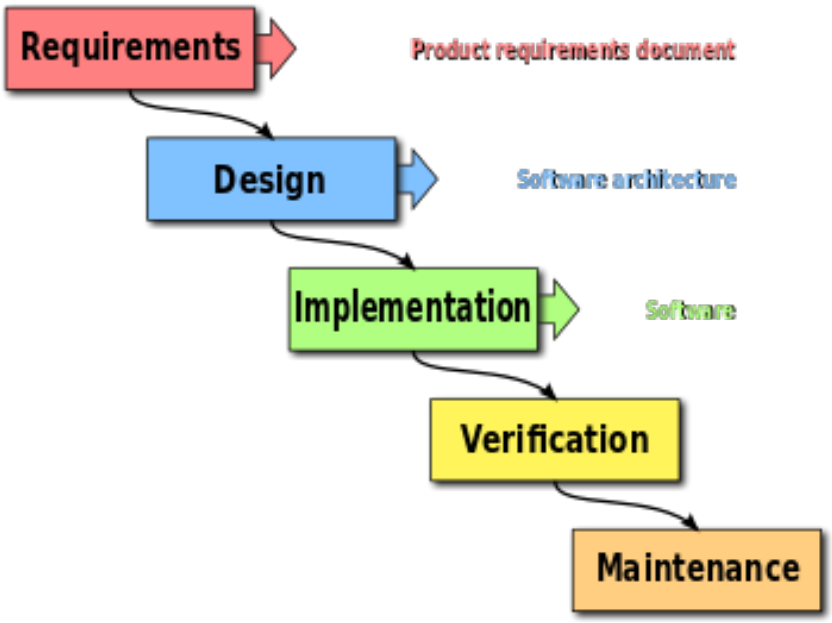


Figura 2.1: Modelo en cascada.

- El proyecto está dividido en fases secuenciales.
- Se hace hincapié en la planificación, los horarios, fechas, presupuestos y ejecución de todo un sistema de una sola vez.

- Se mantiene un estricto control durante la vida del proyecto a través de la utilización de una amplia documentación escrita.

2.4. Plan de trabajo

Para poder abordar el problema se han marcado una serie de objetivos a completar. Dichos hitos son los siguientes:

1. Estudio, comprensión y montaje del OpenROV 2.8.
2. Una vez terminado el montaje, se realizará la conexión del ROV para comunicarse con el portátil.
3. **Fase de pruebas**, se probará el robot en distintos entornos para comprobar su funcionamiento.
4. Estudio y comprensión de los paquetes de ROS roslibjs[1] y rosbridge[2]
5. Integración de ROS y OpenROV.

[1] http://wiki.ros.org/roslibjs
[2] http://wiki.ros.org/rosbridge_suite

Capítulo 3

Plataforma OpenROV

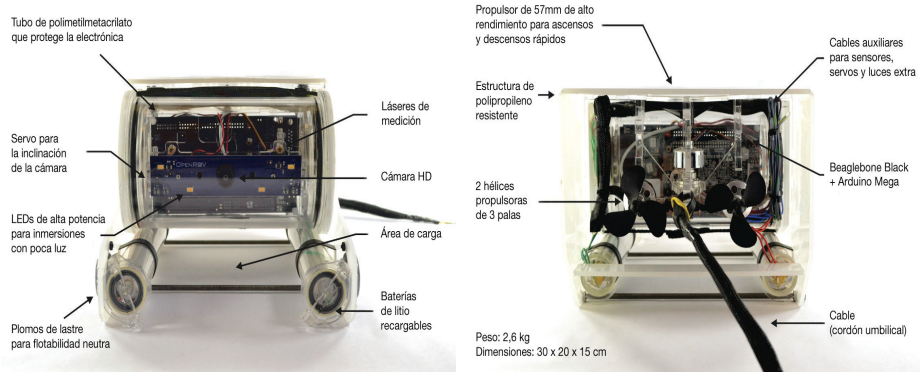
3.1. Robot Submarino OpenROV

La robótica submarina se lleva desarrollando desde mediados de los años sesenta. Los robots submarinos costaban millones, además de ser poco manejables, ya que tenían un tamaño similar al de un coche.
OpenROV [5] es un robot con una cámara que se retransmite en directo a una página web. Su principal virtud es que estos robots funcionan con código abierto.

En general, sus características entran dentro de los parámetros presentados a continuación:

- **Cámara web HD**, transmite un vídeo en alta definición a un portátil a través de un cable de par trenzado.
- **Tres servomotores**, utilizados para maniobrar el robot.
- **Flotabilidad y manejo**.
- **Interfaz web de usuario**, en ella se puede manipular la cámara, los láser, focos y motores. Una de las principales ventajas de la web, es que no necesita conexión a Internet.
- **Soporte de la comunidad de OpenROV**.

Las siguientes imágenes muestran el OpenROV 2.8 en su parte frontal y trasera.



(a) OpenROV parte frontal (b) OpenROV parte trasera

Figura 3.1: OpenROV

3.2. Estructura y Montaje

En esta sección se va a describir cada paso del montaje del robot, desde el montaje de cada uno de los chasis hasta la electrónica, de la cual tendremos que soldar los elementos.

3.2.1. Estructura de OpenROV

Antes de comenzar con el montaje de la estructura, la cámara y el chasis del ROV, se necesitarán ciertas herramientas para el correcto ensamblaje.

Las herramientas necesarias son: guantes, gafas, cúter, cemento acrílico y pegamento. Una vez hayamos obtenido todos los materiales empezaremos con el montaje.

Figura 3.2: Piezas de los chásis.

Se separan las piezas para realizar su correcto montaje. Una vez localizadas las piezas para la estructura, se le quitará el papel protector a la pieza. Iremos pegando y dejando secar para que se adhiera correctamente. Al final, quedará una estructura como la de la imagen.

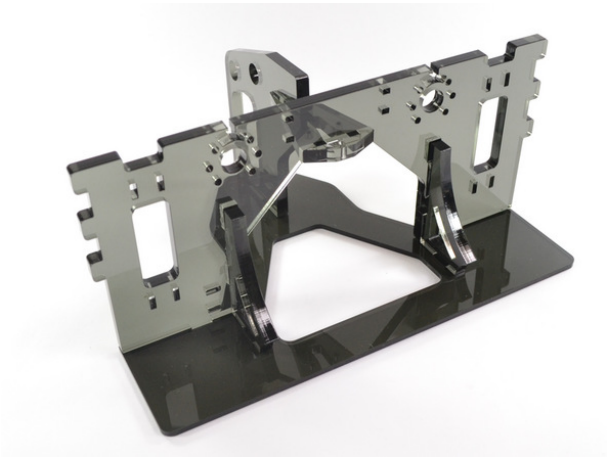
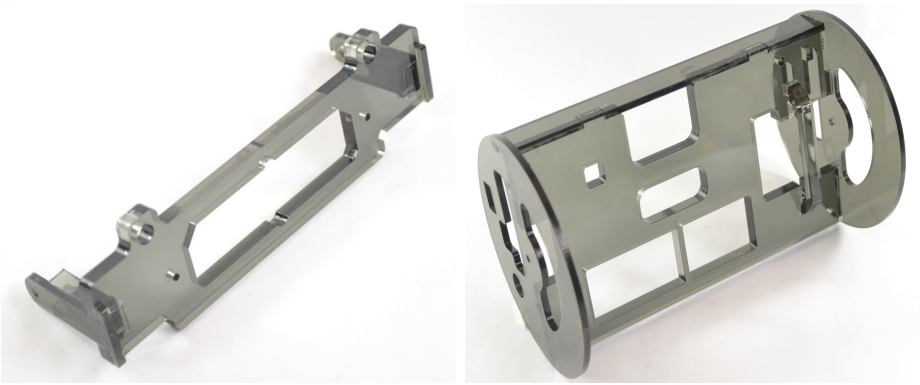


Figura 3.3: Chasis principal.

Para el montaje de la estructura del soporte de la cámara y el chasis de la electrónica se realizará el mismo procedimiento que en el caso anterior, con un resultado como el de las siguientes fotos:



(a) Soporte de la cámara (b) Chasis de la electrónica

Figura 3.4: Soporte de la cámara y electrónica

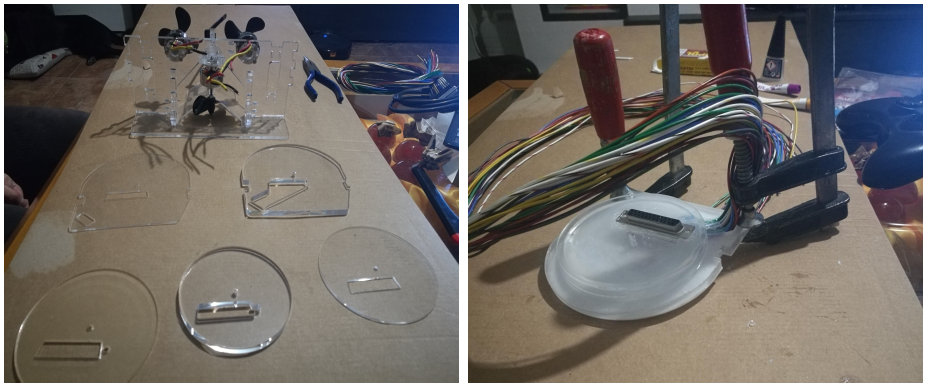
3.2.2. Tapas del chásis

Al igual que en el punto anterior, antes de comenzar con el montaje de la tapa, se necesitarán ciertas herramientas para el correcto ensamblaje.

Las herramientas necesarias son: guantes, gafas, cúter, cemento acrílico, SuperGlue y epoxy (epoxy es un tipo de pegamento que soporta estar en contacto con el agua). Una vez hayamos obtenido todos los materiales empezaremos con el montaje.

En este caso, existen dos tapas. En la primera (la superior), necesitaremos una jeringa proporcionada por la comunidad de OpenROV, la cual utilizaremos para preparar la impermeabilización y el sellado de los componentes software del robot.

En la segunda tapa, está el conector DB-25, el cual, conecta todos los componentes eléctricos. Una vez realizado el montaje de la tapa, se pegará en el espacio sobrante el conector DB-25 y se colocarán los cables adecuadamente en el hueco de la tapa. Cuando esté todo debidamente colocado se pegará la tapa superior al montaje realizado, lo cual dejará todo debidamente sellado.



(a) Tapa inferior desmontada (b) Tapa inferior montada

Figura 3.5: Tapa inferior

3.2.3. Electrónica

Este es el punto crucial del funcionamiento del robot y se necesita especial atención en cada uno de los pasos.

Al igual que en los apartados anteriores, necesitaremos material especial, entre los que están el soldador, estaño, PLC, BeagleBone, tijeras y destornilladores.

Para empezar, desmontaremos los PLC para sacar el hardware, el cual utilizaremos más adelante. Uno de los PLC estará interno en el robot, y el otro, será el extremo conector que comunicará el ROV con el portátil.

Continuaremos con la placa BeagleBone, que es el procesador principal en el ROV. Funciona junto con un Ardunio para controlar la mayoría de las funciones del ROV.

Una vez juntos el PLC, el BeagleBone y la placa del ROV (Raspberry Pi), constituirán el hardware del robot. Estos elementos se atornillaran para obtener una mejor sujeción. El resultado final lo podemos observar en la siguiente imagen.

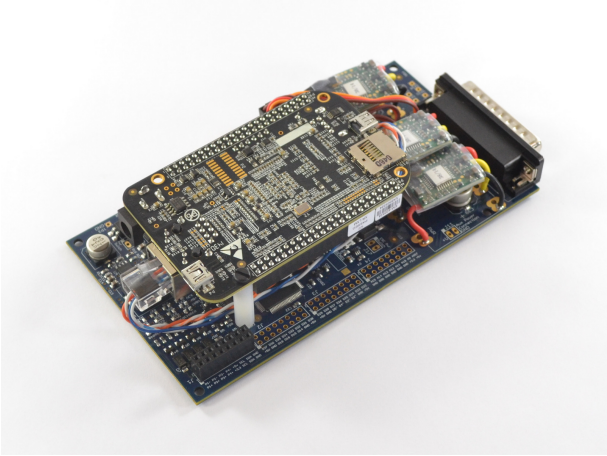


Figura 3.6: Raspberry Pi + PLC + BeagleBone

Después, conectaremos el servo al chasis, el cual va a servir para mover la cámara verticalmente.

Continuaremos con la soldadura de los láser y unos cables a la placa de video del ROV.

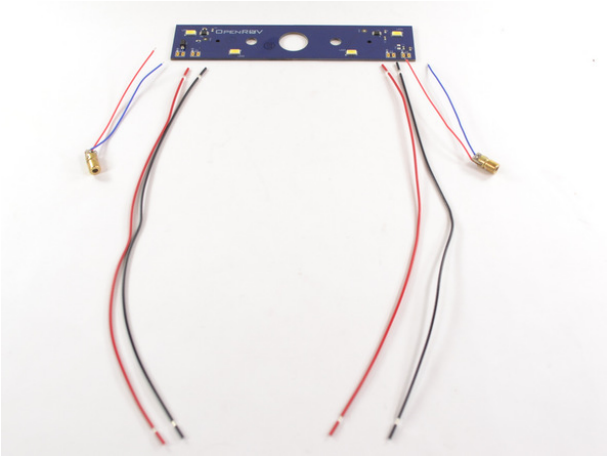


Figura 3.7: Electrónica de la cámara

Cuando estén debidamente soldados, se colocará el chasis de la cámara, junto con lo que acabamos de soldar y la cámara. A estos tres elementos los juntaremos con tornillos y tuercas. Para finalizar con este paso, comprobamos que hay un conector al cual enchufaremos un cable que irá directo a la placa principal del ROV.

Para la sujeción de la cámara utilizaremos el chasis principal, al que atornillaremos para fijar la cámara.

Para finalizar este apartado, se conectará el hardware del robot en el lado opuesto de la cámara, y se fijará con tornillos. Se realizará la conexión de los cables a sus ranuras correspondientes.

3.2.4. Enrutamiento

En esta fase, se va a requerir unas herramientas más específicas para realizar el montaje, ya que necesitaremos soldar e impermeabilizar con una pistola de calor, además, de tijeras, epoxy, super Glue y bridas.

Empezaremos calculando la distancia de los cables y separaremos los de color verde claro y naranja, para la conexión de las baterías. Los cables verdes oscuros, rojos y azules, los utilizaremos para el montaje de los servomotores. Iremos fijando los cables con las bridas para una correcta colocación.

Cada uno de los tres motores servo, lleva tres cables, los cuales soldaremos con los cables correspondientes. Una vez soldados, con una silicona especial se rellenará el espacio del cable soldado y con la pistola de calor se fijará e impermeabilizará, podemos ver parte del proceso en la siguiente imagen.

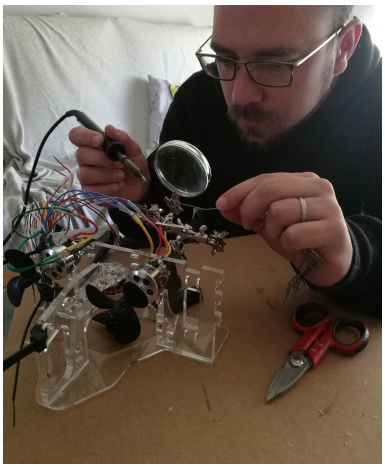


Figura 3.8: Soldadura de cables

Continuaremos con el montaje de los dos tubos de la batería. Primero, necesitaremos realizar el montaje de las piezas y soldaremos el muelle (borne) con un chip facilitado por la comunidad de ROV. Esta será la parte negativa de las baterías. En el extremo del cable sobrante, se rellenará con estaño y se soldará al cable formando la parte positiva en la irá conectada la batería.

3.2.5. Final

En la parte final del montaje, será necesaria una soldadura con el cable de par trenzado que es el encargado de mandar la señal al portátil.

El otro extremo del cable de par trenzado irá conectado al PLC. Gracias a este cable, el robot puede sumergirse a una distancia de 100m. Usaremos una brida de mayor tamaño para fijar el cable y así obtener una mayor seguridad, ya que si el robot quedara expuesto a un problema en la inmersión, es probable que necesitaramos tirar de él.

En el cuerpo principal del robot, con unas gomas sujertaremos los tubos que llevarán las baterías y fijaremos los extremos con dos varillas de metal.
Para aislar el hardware se usará un tubo cilíndrico trasparente de polimetilmetacrilato previamente limado en los extremos.
Cuando todo el montaje esté terminado, pondremos más seguridad al hardware, añadiendo una cuerda que hará presión.

Además, colocaremos dos pesas de 300 gr. en los brazos del ROV para que la sumersión sea más fácil y fiable.

3.3. Conectividad OpenROV

En esta sección se marcarán las pautas necesarias para realizar el correcto funcionamiento del software del OpenROV.

3.3.1. Diseño de Conectividad

Para encender el ROV, necesitaremos el cable mini USB y el cable ethernet debidamente colocado en el extremo del PLC y estos a su vez al portátil.

Figura 3.9: PLC externo

El conector USB actúa como interruptor de encendido/apagado del robot.

El ROV tiene una dirección IP estática 192.168.254.1. Para conectarnos a la web del robot, la dirección ethernet de nuestro ordenador debe pertenecer a la misma subred, es decir, 192.168.254.2. La máscara de subred se establece en la dirección 255.255.255.0.

Pero para poder mostrar la página, antes debemos realizar una serie de pasos, en la que necesitaremos una tarjeta microSD para realizar la configuración del ROV.

1. Se descarga la imagen más reciente de OpenROV 2.8, en este caso, la versión es 30.0.3. Una vez descargado, se descomprime el archivo .zip, dejando en el directorio de descarga un ficherom con la extensión .img.

2. Formateamos la tarjeta microSD, y montaremos la imagen descargada en la tarjeta. Una vez terminado el proceso, quitaremos la tarjeta del portátil.

3. Conectamos la tarjeta microSD a el BeagleBone, y conectamos el ROV con el cable USB para encenderlo. Los LED de la placa Beagle Bone parpadearán para mostrar que la imagen se está aplicando a la memoria de la placa. El proceso termina cuando los cuatro led se queden encendidos, si pasado un tiempo los cuatro leds no lucen, habría que repetir este punto.

   Cuando esté completo, retiraremos el cable USB del portátil para desconectar el ROV y retiraremos la tarjeta microSD de la placa Beagle Bone.

   Terminado este proceso, volveremos a conectar la tarjeta en el portátil.

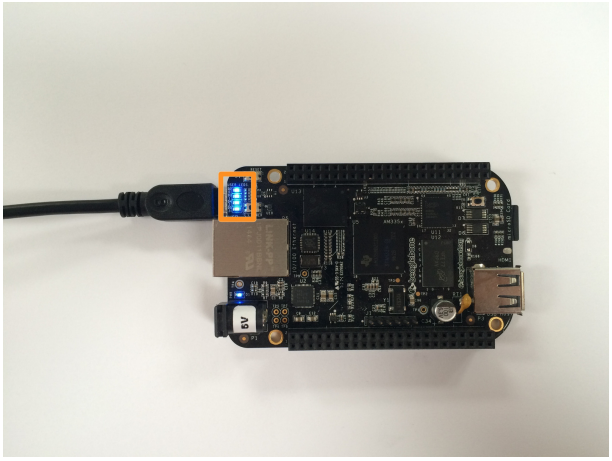


Figura 3.10: BeagleBone configurada

4. Como se mencionó al principio del apartado, el ROV tiene una dirección IP estática incorporada de 192.168.254.1. Para conectarnos, configuraremos nuestro ordenador con la dirección IP 192.168.254.2 y con la máscara de subred en la dirección 255.255.255.0.

   En Windows 8 la configuración sería:

Configuración 3.1: Windows 8

```
Ir  al Panel de control.
Seleccionar Red e Internet.
Red y centro compartido  (click) cambiar configuracin de adaptador
   (click) ethernet.
Propiedades  (click) usar la siguiente direccin IP
Insertarla direccin IP 192.168.254.2
```

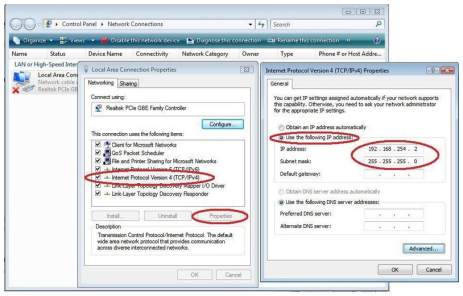


Figura 3.11: Conexión Windows 8

En Ubuntu la configuración de una nueva IP estática:

Configuración 3.2: Ubuntu

```
# Abrir el terminal (Ctrl + Atl + T).
# Comprobamos las interfaces del equipo:
$ ifconfig -a
# Editamos el archivo de configuracion de las interfaces:
$ sudo vim /etc/network/interfaces

# Configuracion de direccion IP fija para el interfaz eth0
  auto eth0
  iface eth0 inet static
  address 192.168.254.2
  netmask 255.255.255.0
  network 192.168.254.0
  broadcast 192.168.254.255
  gateway 192.168.254.1
```

Donde:

address es la dirección IP que quieres ponerle a tu máquina.

netmask es la máscara de subred de esa dirección IP.

network es la red a la que pertenece esa dirección IP.

broadcast es la direción IP de difusión de esa red.

gateway es la dirección IP de la puerta de enlace predeterminada.

Reiniciamos las interfaces de red para aplicar los cambios:

3.3: Reinicio

```
$ sudo /etc/init.d/networking restart
```

5. Abriremos el navegador Google-Chrome, y en la barra de direcciones escribiremos la IP 192.168.254.1:8080, que es la dirección IP de OpenROV. Accederemos a la web, que es la interfaz de OpenROV.

6. Para finalizar, deberemos actualizar el firmware, en la web, presionaremos el botón de configuración en la parte superior derecha de la pantalla y presionaremos "Cargar firmware desde la tarjeta SD a Ardruino" La tarjeta microSD no debe estar en el ROV, presionaremos "Mostrar Detalles" y seguidamente, "Aplicar nuevo Firmware". Una vez que la carga sea exitosa, como el de la imagen, se retirará el cable USB del portátil para desconectar el ROV. Volveremos a conectar el cable USB al portátil para realizar el reinicio.

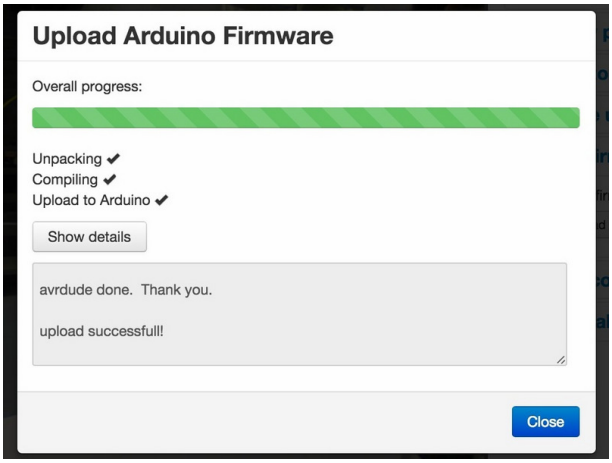


Figura 3.12: Actualización firmware

3.3.2. Puesta a punto de los elementos

Cámara

Ajustaremos el foco de la lente de la cámara hasta que la imagen pueda verse con nitidez.

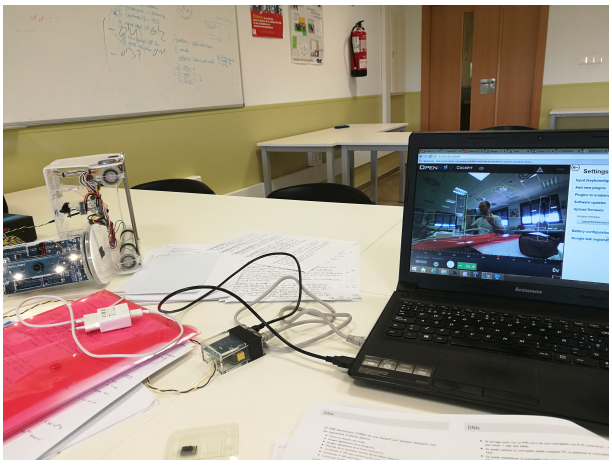
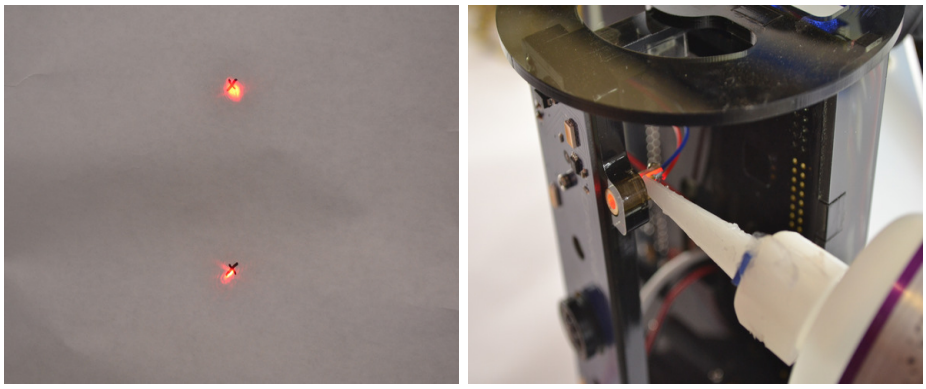


Figura 3.13: Foco cámara

Láser

El ROV lleva incorporado dos láser, que hemos soldado previamente a la placa de la cámara. El proceso que llevaremos a cabo será su calibración y pegado en el chásis. En un papel dibujaremos dos "X" separadas entre si diez centímetros, y colocaremos esa hoja a una distancia de entre tres y cuatro metros del robot, el cual, debe estar a la misma altura que el folio. Si pesionamos la tecla L en la web, encenderemos los

láseres. Para apagarlos, simplemente hay que volver a pulsar la tecla L. Una vez este todo calibrado, pegaremos con Super Glue los láseres al chasis de la cámara.



(a) Distancia de los laseres (b) Fijado del los laseres

Figura 3.14: Láser

Motores

En este punto, se realizará una comprobación de las hélices. Para ver si su rotación es la correcta presionaremos la tecla de "flecha arriba", y observaremos en que dirección giran los motores. El motor izquierdo (amarillo) debe funcionar en sentido contrario a las agujas del reloj. El motor derecho (azul) debe funcionar en sentido a las agujas del reloj.

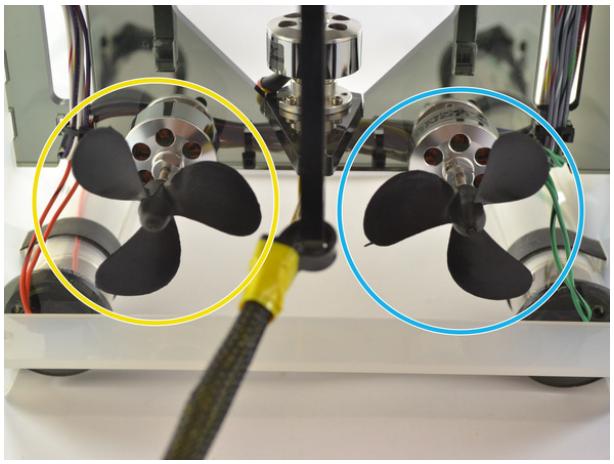


Figura 3.15: Motores

Si el giro es incorrecto, invertiremos su rotación en la parte de "diagnóstico" de la web. Seleccionaremos la opción "invertir" del motor o motores que queramos modificar.

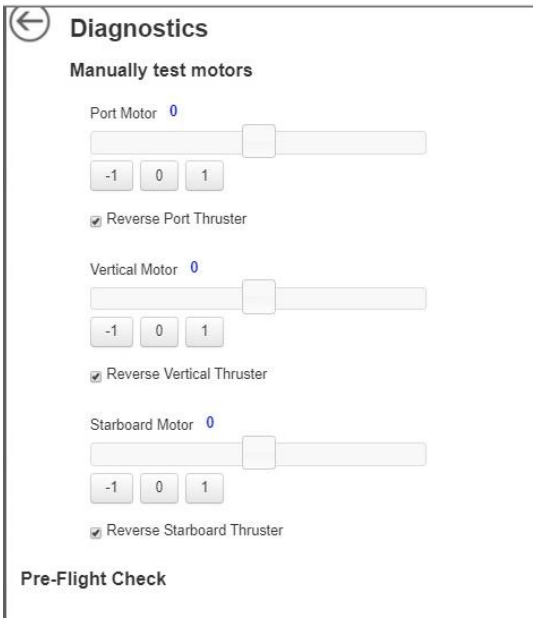


Figura 3.16: Diagnostics

Tubo cilíndrico trasparente de polimetilmetacrilato

El tubo cilíndrico contendrá todo el sistema hardware y software del ROV. Para una mejor fijación, lijaremos los bordes interiores para que la junta se ajuste al tubo.

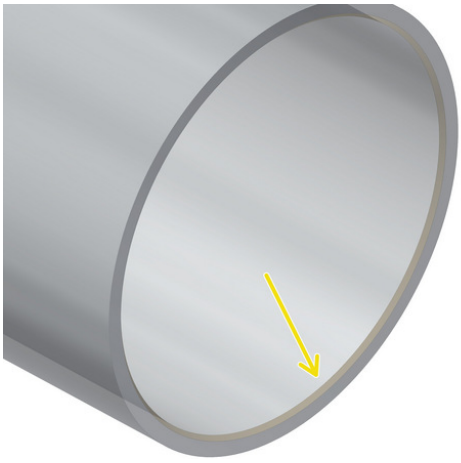


Figura 3.17: Tubo

Tapa superior

Una vez instalado el tubo, verificamos de que la junta esté bien introducida a lo largo de todo el perímetro interior del tubo. La tapa superior contiene un orificio de ventilación para igualar la presión al colocar las tapas.

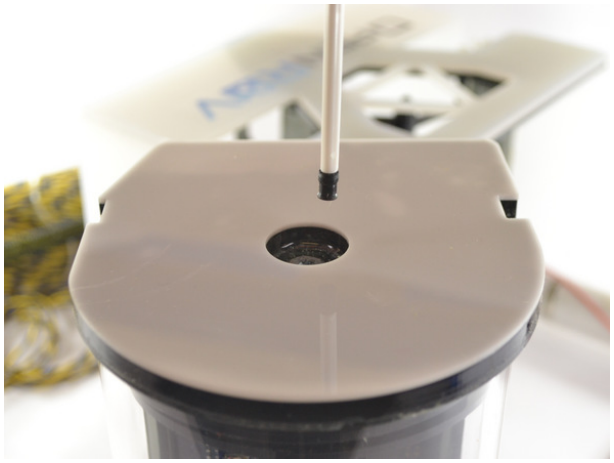


Figura 3.18: Tapa superior

3.3.3. Web/Cockpit

El manejo del ROV se consigue gracias a la web Cockpit, que es la interfaz de usuario y contiene todos los sistemas de control de cualquier vehículo o dispositivo operado a distancia.

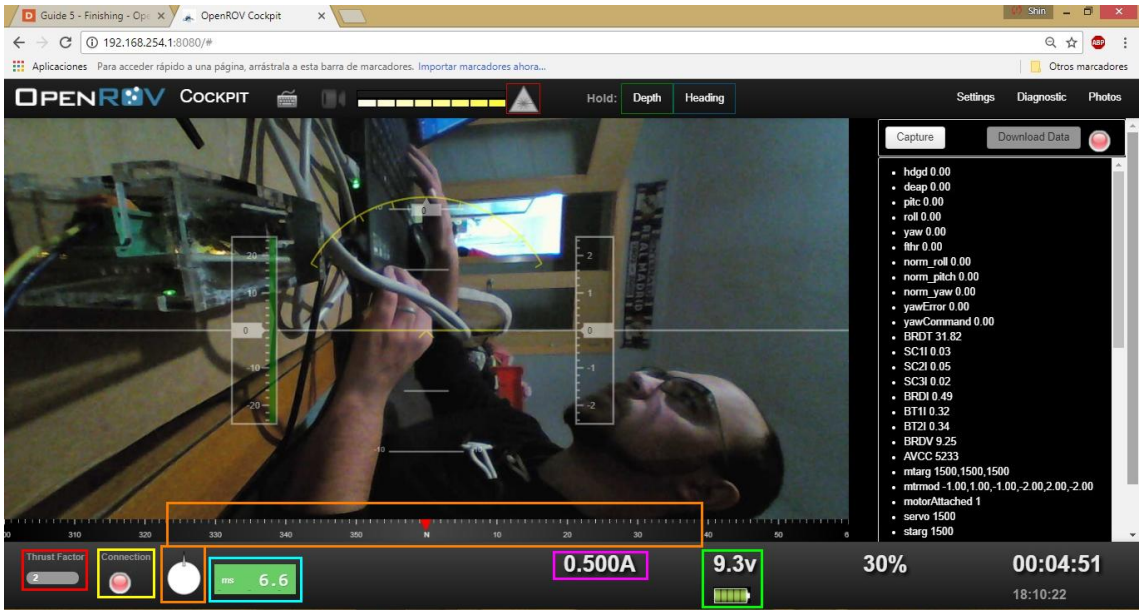


Figura 3.19: Cockpit (1)

- Configuración de potencia de los motores (1, la más baja y 5, la más alta).
- Conectividad (verde = conectado).
- Brújula.

- Latencia (si hubiera algún problema mostraría +999).
- Intensidad de corriente los componentes electrónicos en Amperios.
- Potencia de las baterías.

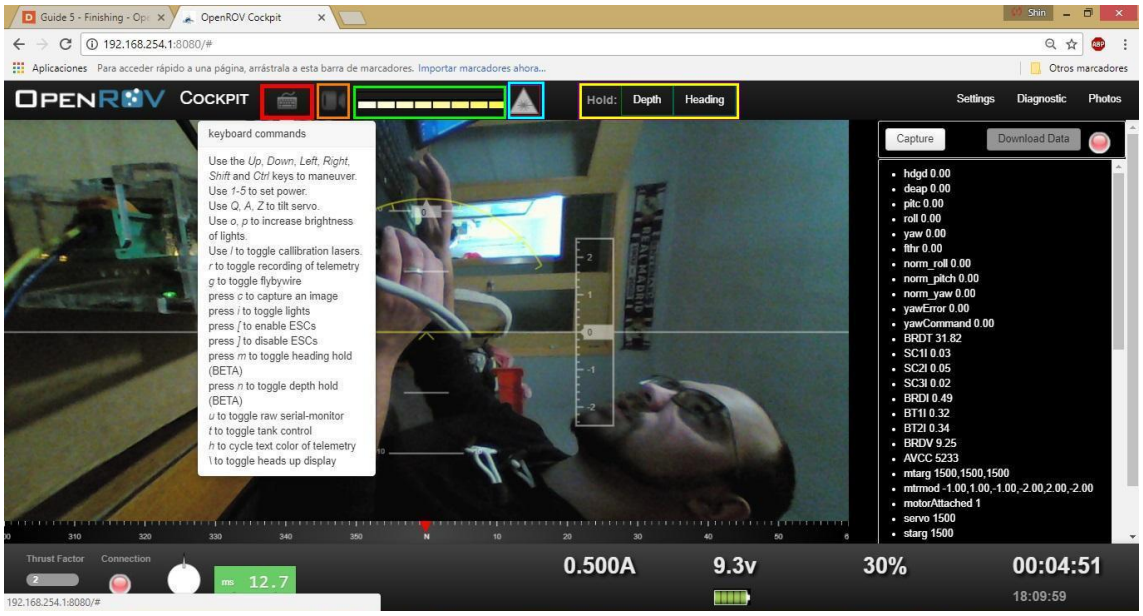


Figura 3.20: Cockpit (2)

- Pulsando el botón señalado en rojo (el icono del teclado) se obtienen los comandos de uso del robot.
- Indicador de la cámara.
- Indicador de nivel de luz.
- Indicador de encendido/apagado del láser.
- Indicadores de profundidad y rumbo. Se alternan entre activo/desactivo para mantener la profundidad actual y/o el rumbo.

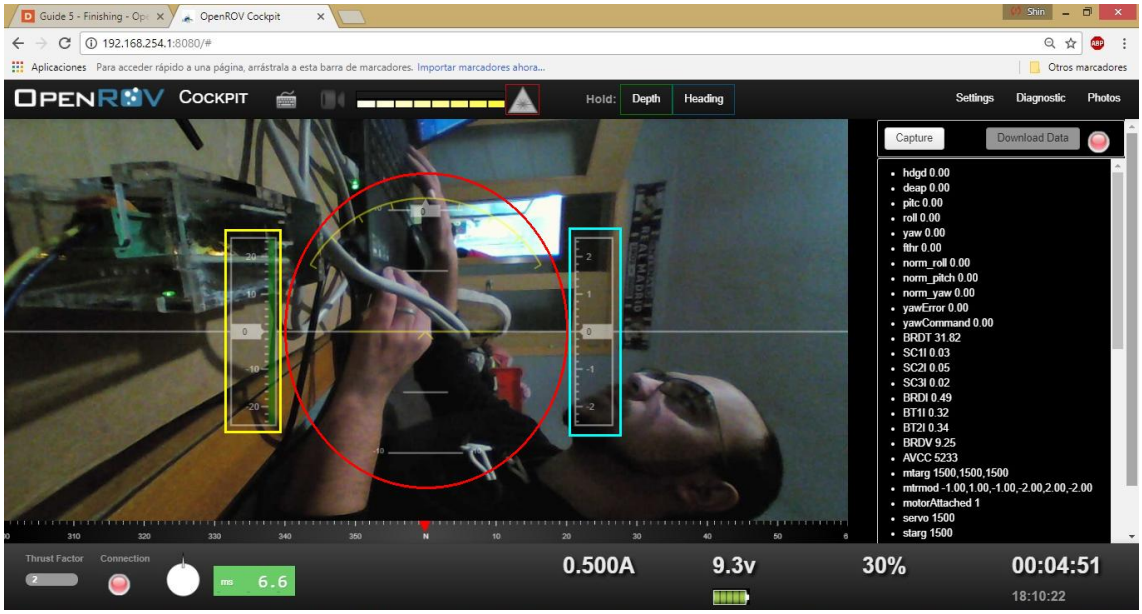


Figura 3.21: Cockpit (3)

- Fuerza de empuje del motor.
- Horizonte artificial.
- Profundidad.

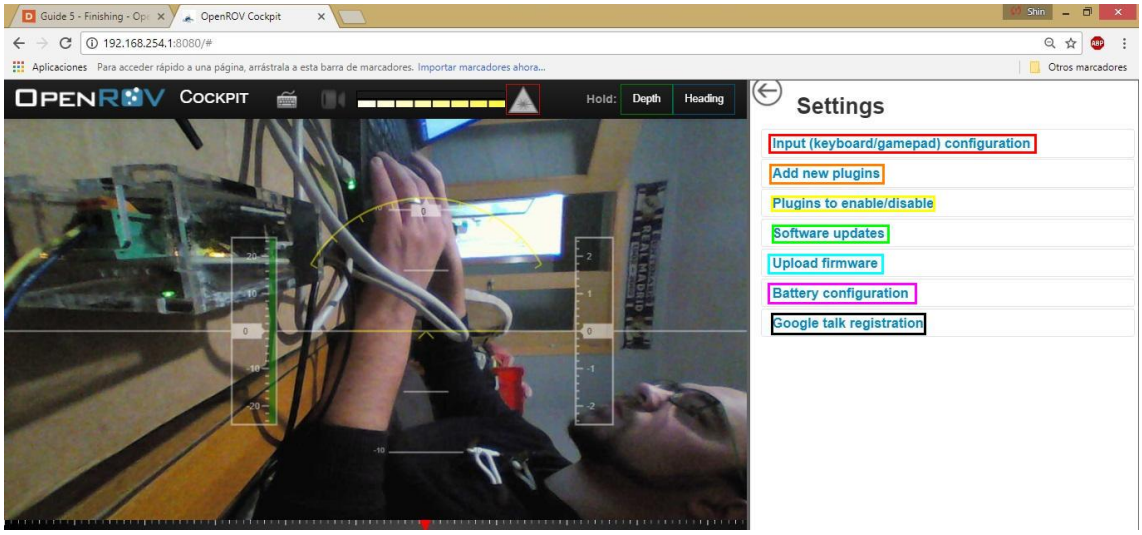


Figura 3.22: Ajustes

Panel de configuración:

- Configuración de entrada: personalizar las comandos de uso del robot.
- Agregar nuevos complementos.
- Habilitar/deshabilitar complementos: activa y desactiva los complementos agregados en el punto anterior.
- Actualización de software.
- Actualización del firmware.
- Configuración de la batería: Se establece el voltaje mínimo y máximo de la batería.
- Registro de Google Talk: Permite el acceso a las de redes sociales de la web.

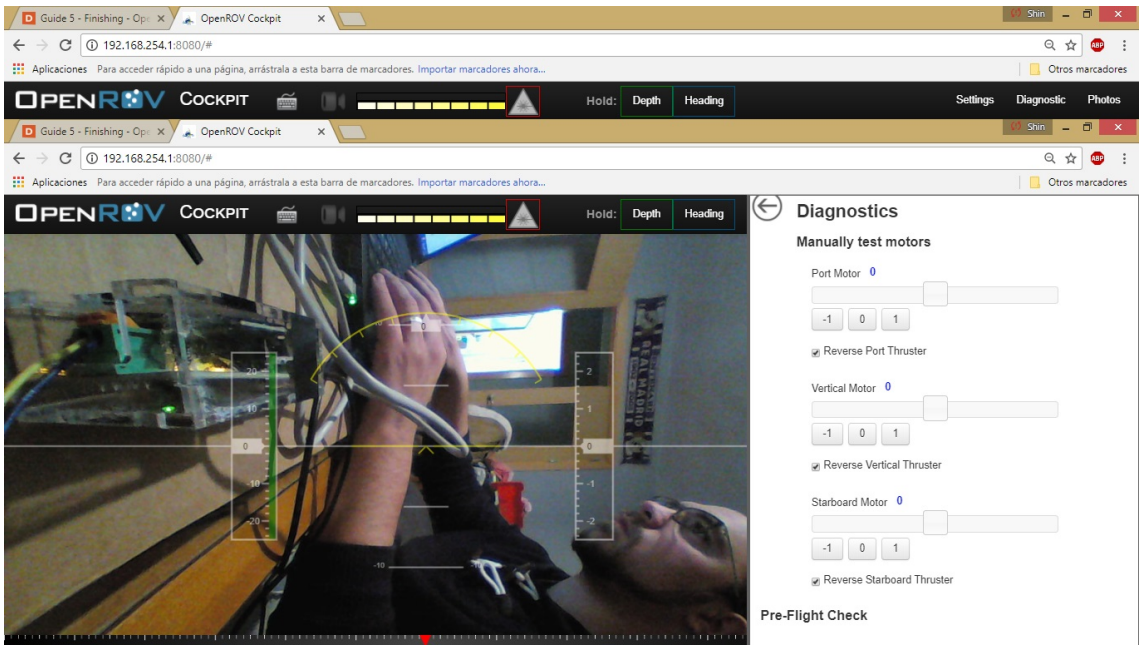


Figura 3.23: Diagnóstico

Panel de diagnósticos:

- Verificación manual los motores.
- Si hay un error con la dirección de los motores se puede invertir desde este panel.
- Pre-Flight Check, es el marcador de la posición actual.

El panel de configuración de OpenROV se encuentra en: 192.168.254.1
Hay dos partes:

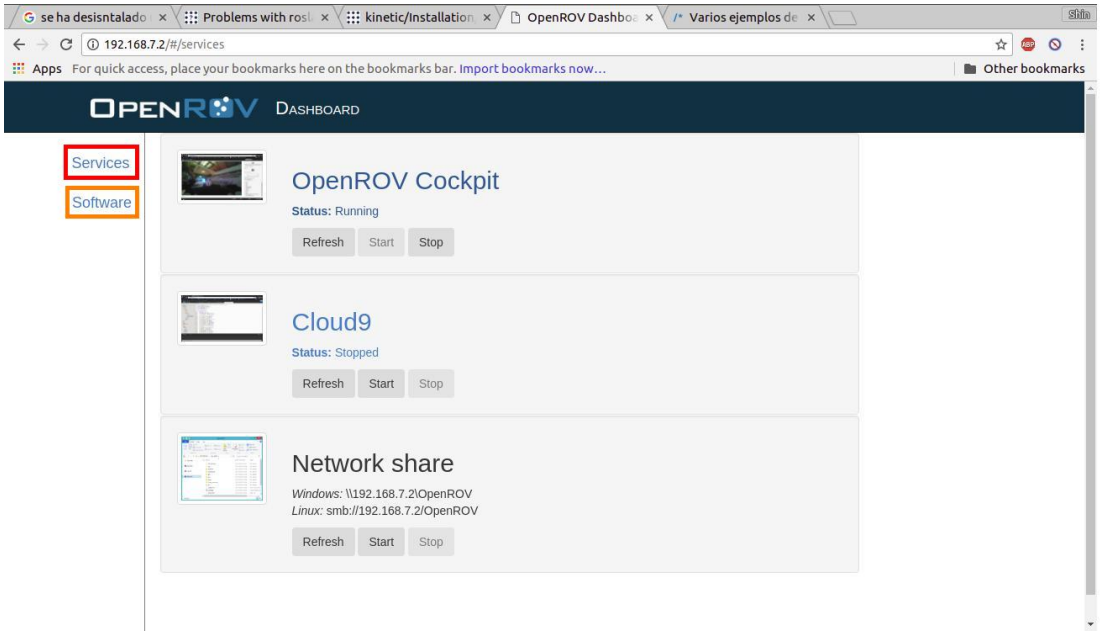


Figura 3.24: Servicios

- **Servicios**: Reiniciar la web del ROV.
- **Software**: Determinar la versión del software y puede ejecutar la actualización del software del ROV.

3.3.4. Mantenimiento

Para que la vida útil del robot sea lo más larga posible, se deben seguir unas ciertas pautas de mantenimiento.

Enjuagar

Una vez que hayamos utilizado el robot, tendremos que enjuagarlo en agua después de cada inmersión, para mantenerlo limpio y libre de desechos y minerales.

Mantenimiento de los motores

Después de haber utilizado el robot en una inmersión, se debe limpiar los motores, ya que tienden a corroerse fácilmente. Se recomienda lubricarlos. Los pasos que debemos seguir son los siguientes:

1. Enjuagar los motores en agua fría.
2. Secar los motores (Es recomendable usar una pistola de aire comprimido).
3. Para finalizar el mantenimiento, lubricaremos los motores con un spray.

Estos puntos deben hacerse antes y después de cada inmersión.

Secado

Antes de guardar el OpenROV en su maletín, debemos asegurarnos de que no queda ninguna pieza mojada. Para un mayor secado utilizaremos una pistola de aire comprimido y desmontaremos los tubos de las baterías y el tubo principal para cerciorarnos de que las baterías y la electrónica están secos.

Carga de baterías

El robot funciona con seis baterías, las cuales se cargarán a la vez, ya que se necesitan todas para el funcionamiento del robot.
Cada tubo lleva tres baterías. Si se intuye que alguna tiene un rendimiento inferior, se verificará utilizando un polímetro.
Es importante mantener las baterías secas y sin corrosión

3.3.5. Problemas

Durante la vida útil del OpenROV pueden surgir ciertos imprevistos, a continuación se detallan los problemas más comunes y como solucionarlos.

1. **Vaho** durante todo el proceso de sumersión.

   El tubo principal puede empañarse tanto en la superficie (días soleados) como durante la sumersión.

   Si el problema ha surgido en la superficie, retiraremos el tubo principal y lo limpiaremos con un paño. Si se empaña durante la inmersión, lo más recomendable es salir a la superficie y verificar que no existe ninguna fuga en el tubo principal.

   La forma de solucionar el vaho es a través de los productos anti-vaho, además de cerciorarnos de que el tubo es estanco.

2. **Vídeo**.

   Otro de los problemas más comunes es un corte o apagón en el vídeo que reproduce el robot en el fondo oceánico.

   Para evitar este fallo, es recomendable actualizar periódicamente la imagen del software en el BeagleBone.

3. **Control u otros problemas misceláneos**.

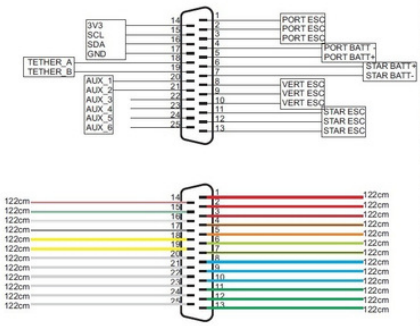
   En las situaciones en las que el error provenga del control o fallos ajenos, el procedimiento a seguir es reiniciar el robot desenchufando el clable USB del PLC y volver a conectarlo.

4. **Pérdida aleatoria de energía de la batería**.

   Si comprobamos que existen pérdidas de energía, primero comprobaremos que las baterías hacen contacto entre si y que están colocadas de la forma correcta.

   Si el fallo persiste, se verifica la potencia (alrededor de los 9V) a través de los pines correspondientes del conector DB-25.

   Además, se puede comprobar cada batería individual para asegurarse de que tenga el voltaje correcto a través de un polímetro.



PIN	FUNCTION	COLOR	LENGTH
1	PORT ESC	RED	122CM
2	PORT ESC	RED	122CM
3	PORT ESC	RED	122CM
4	PORT BATTERY -	BLACK/ORANGE	122CM
5	PORT BATTERY +	ORANGE	122CM
6	STARBOARD BATT +	LIGHT GREEN	122CM
7	STARBOARD BATT -	BLACK/LIGHT GREEN	122CM
8	VERT ESC	BLUE	122CM
9	VERT ESC	BLUE	122CM
10	VERT ESC	BLUE	122CM
11	STARBOARD ESC	GREEN	122CM
12	STARBOARD ESC	GREEN	122CM
13	STARBOARD ESC	GREEN	122CM
14	3V3	GRAY/RED	122CM
15	SCL	GRAY/GREEN	122CM
16	SDA	GRAY/WHITE	122CM
17	GND	GRAY/BLACK	122CM
18	TETHER_A	YELLOW	122CM
19	TETHER_B	YELLOW	122CM
20	AUX_1	GRAY	122CM
21	AUX_2	GRAY	122CM
22	AUX_3	GRAY	122CM
23	AUX_4	GRAY	122CM
24	AUX_5	GRAY	122CM
25	AUX_6	GRAY	122CM

Figura 3.25: DB-25

5. **Enredos**.

   Facilmente el robot puede quedar atrapado en el suelo oceánico por culpa de las algas u otros objetos o vegetales.

   Lo mejor es evitar pilotar el robot en un paisaje con una vegetación pesada.

   Si comprobamos que no se puede sacar el OpenROV por medio de los comandos, tiraremos de la correa hasta la superficie. Aunque puede existir un momento en el que el robot no pueda ser recuperado por ese medio, en ese caso, se puede cortar la correa e ir a buscar al robot, ya que se puede volver a soldar la correa.

6. **Inundación del tubo principal**.

   El mayor problema al que nos podemos enfrentar, es que se inunde el tubo principal, que conllevaría al riesgo de perder el hardware del robot.

   Primero, se desconecta la alimentación del robot desenchufando el cable USB y seguidamente retiraremos las baterías.

   Enjuagamos los componentes electrónicos que se mojaron en el agua para eliminar los residuos del hardware, desmontaremos los componentes electrónicos y volveremos a enjuagar.

   Secaremos bien cada una de las piezas de la electrónica. Lo más recomendable es usar una pistola de aire comprimido.

   Para finalizar, se colocaran todos los componentes electrónicos afectados en un lugar seco. Lo mejor es usar una bolsa de llena de arroz.

   Cuando se vuelve a montar el robot, verificaremos que no exista corrosión. Si la hubiera, se intentará eliminar con un cepillo.

Capítulo 4

Integración de ROS en OpenROV

En este capítulo se va a describir como se ha realizado la integración de ROS con OpenROV [1].

Para ello, utilizaremos las bibliotecas de ROS Rosbridge y Roslibjs, que nos proporcionarán la comunicación entre el ordenador y el robot OpenROV.

4.1. Configuración del ROV

La electrónica de OpenROV se basa en un Arduino conectado a una placa BeagleBone, que ejecutará el software de OpenROV.

Para la implementación de ROS con OpenROV se ha decidido implementar ROS en el portátil en lugar de en el ROV, ya que se necesita ejecutar un ROS Master. Si se hubiera realizado a la inversa, se necesitaría acceso al robot para ejecutar el ROS Master, por lo que una vez sumergido no podríamos utilizar ROS.

El ROS Master se ejecutará en el portátil y será el que interactue con los nodos de ROS instalados, en este caso, Rosbridge.

Para la comunicación de ROS y OpenROV usaremos el paquete Roslibjs el cual instalaremos dentro del software del robot. Roslibjs es la biblioteca que utiliza JavaScript para comunicarse con ROS. Roslibjs publicará y/o suscribirá a los mensajes de ROS.

Se utilizará el paquete de Rosbridge para traducir los mensajes JSON en mensajes ROS (OpenROv utiliza JSON para su comunicación). Este paquete actúa como un Nodo de ROS que puede publicar y/o suscribir a los mensajes.

En conclusión, la comunicación se realizara a través de una conexión Websocket (Rosbridge) y el paquete de Roslibjs.

La figura a continuación muestra la estructura básica de ROS para la integración de OpenROV.

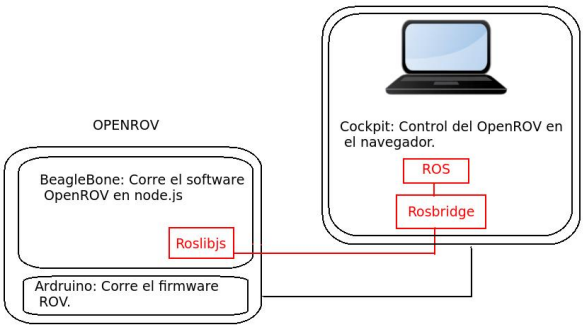


Figura 4.1: Diagrama de la conexión ROS y OpenROV

Para configurar el robot se debe instalar el Roslibjs, sin embargo, este paquete necesita de varias dependencias.

Comenzaremos configurando el BeagleBone a través de una conexión SSH, conectando un cable Ethernet entre el BeagleBone y el router. Para activar el hardware usaremos un cable miniUSB conectado entre el BeagleBone y el portátil. Una vez realizadas las conexiones, haremos un SSH a la placa.

4.1: SSH

```
$ ssh rov@192.168.7.2
```

cuando pida la contraseña, deberemos introducir "OpenROV".

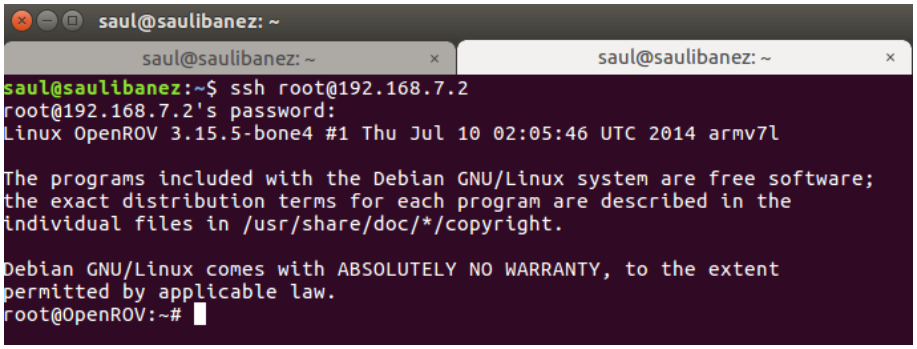


Figura 4.2: SSH

Para instalar las dependencias de Roslibjs, ejecutaremos los siguientes comandos:

4.2: Dependencias Roslibjs

```
sudo apt-get update
sudo apt-get install libcairo2-dev libjpeg-dev libpango1.0-dev libgif-dev
    build-essential g++
```

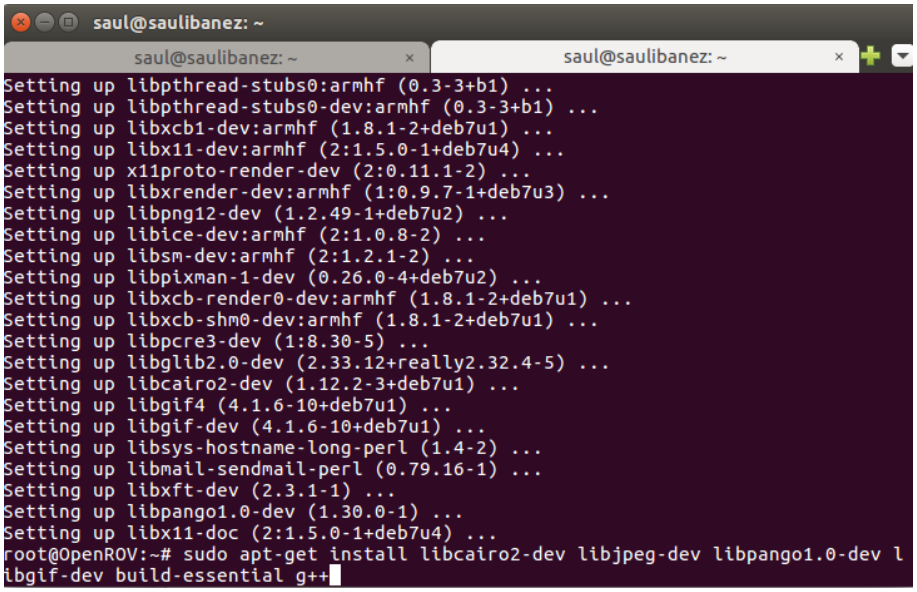


Figura 4.3: Dependencias Roslibjs

Continuaremos instalando otras dependencias de Roslibjs, en este caso, npm, que es un gestor de paquetes que utiliza Node.js.

Para que no surjan errores con el proxy, debemos deshabilitarlo y realizar un reseteo del sistema.

4.3: Dependencias Roslibjs

```
$ sudo /etc/init.d/openrov-proxy stop
$ sudo /etc/init.d/openrov restart
$ sudo chown -R $USER /usr/local
$ npm install canvas
```

```
saul@saulibanez: ~
saul@saulibanez: ~        saul@saulibanez: ~
root@OpenROV:~# cd /opt/openrov/cockpit/src/plugins
root@OpenROV:/opt/openrov/cockpit/src/plugins# npm install canvas
npm http GET https://registry.npmjs.org/canvas
npm http 200 https://registry.npmjs.org/canvas
npm http GET https://registry.npmjs.org/canvas/-/canvas-1.6.9.tgz
npm http 200 https://registry.npmjs.org/canvas/-/canvas-1.6.9.tgz
npm http GET https://registry.npmjs.org/nan
npm http 200 https://registry.npmjs.org/nan
npm http GET https://registry.npmjs.org/nan/-/nan-2.8.0.tgz
npm http 200 https://registry.npmjs.org/nan/-/nan-2.8.0.tgz

> canvas@1.6.9 install /opt/openrov/cockpit/node_modules/canvas
> node-gyp rebuild

make: Entering directory `/opt/openrov/cockpit/node_modules/canvas/build'
  SOLINK_MODULE(target) Release/obj.target/canvas-postbuild.node
  SOLINK_MODULE(target) Release/obj.target/canvas-postbuild.node: Finished
  COPY Release/canvas-postbuild.node
  CXX(target) Release/obj.target/canvas/src/Canvas.o
In file included from ../node_modules/nan/nan.h:199:0,
                 from ../src/Canvas.h:22,
                 from ../src/Canvas.cc:7:
../node_modules/nan/nan_maybe_pre_43_inl.h: In function ‘Nan::Maybe<bool> Nan::D
efineOwnProperty(v8::Handle<v8::Object>, v8::Handle<v8::String>, v8::Handle<v8::
Value>, v8::PropertyAttribute)’:
../node_modules/nan/nan_maybe_pre_43_inl.h:159:59: warning: suggest parentheses
around ‘&&’ within ‘||’ [-Wparentheses]
  CXX(target) Release/obj.target/canvas/src/CanvasGradient.o
In file included from ../node_modules/nan/nan.h:199:0,
                 from ../src/Canvas.h:22,
                 from ../src/CanvasGradient.cc:9:
../node_modules/nan/nan_maybe_pre_43_inl.h: In function ‘Nan::Maybe<bool> Nan::D
efineOwnProperty(v8::Handle<v8::Object>, v8::Handle<v8::String>, v8::Handle<v8::
Value>, v8::PropertyAttribute)’:
../node_modules/nan/nan_maybe_pre_43_inl.h:159:59: warning: suggest parentheses
around ‘&&’ within ‘||’ [-Wparentheses]
  CXX(target) Release/obj.target/canvas/src/CanvasPattern.o
In file included from ../node_modules/nan/nan.h:199:0,
                 from ../src/Canvas.h:22,
                 from ../src/CanvasPattern.cc:8:
```

Figura 4.4: Canvas

La última dependencia para el complemento es Roslibjs, lo instalaremos usando:

4.4: Roslib

```
$ npm install roslib
```

Dentro de la carpeta /opt/openrov/cockpit/src/plugins, crearemos un directorio llamado ros (sudo mkdir ros) y en el, se creará un fichero llamado index.js, que ejecutará la conexión con el ROS Master localizado en el portátil.

4.5: Conexión con ROS

```
function ros(name, deps) {
      console.log("ROS plugin started");

      // Comienza la sesion de ROS
      var ROSLIB = require("roslib")
      var ros = new ROSLIB.Ros({
            url : 'ws://192.168.7.1:9090'
      });

      ros.on('connection', function() {
            console.log('ROS connected to websocket');
      });

      ros.on('error', function(error) {
            console.log('ROS error connecting to websocket');
      });

       ros.on('close', function() {
            console.log('ROS closed websocket connection');
      });
   };

   module.exports = ros;
```

También se debe modificar el fichero hosts (sudo vim /etc/hosts) y en el realizaremos los siguientes cambios:

4.6: /etc/hosts

```
#127.0.1.1 OpenROV
192.168.7.2 OpenROV
192.168.7.1 saul.ibanez.local
```

4.2. Configuración de ROS

Para la conexión con ROS, se decide instalar ROS Kinetic en el portátil:

4.7: Instalacion de ROS

```
sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc)
   main" > /etc/apt/sources.list.d/ros-latest.list'
sudo apt-key adv --keyserver hkp://ha.pool.sks-keyservers.net:80 --recv-key
   421C365BD9FF1F717815A3895523BAEEB01FA116
sudo apt-get update
sudo apt-get install ros-kinetic-desktop-full
apt-cache search ros-kinetic
sudo rosdep init
rosdep update
echo "source /opt/ros/kinetic/setup.bash" >> ~/.bashrc
```

```
source ~/.bashrc
sudo apt-get install python-rosinstall python-rosinstall-generator
   python-wstool build-essential
```

Instalaremos Rosbridge

4.8: Rosbridge

```
sudo apt-get install ros-kinetic-rosbridge-server
```

Cuando se configuró ROS, creamos una carpeta llamada catkin_ws, en ella, se creó también la carpeta src, en la que crearemos el directorio openrov el que debe contener un fichero CMakeLists.txt

El archivo CMakeLists.txt es la entrada al sistema de compilación CMake. El código será:

4.9: CMakeLists.txt

```
cmake_minimum_required(VERSION 2.8.3)
project(openrov)

find_package(catkin REQUIRED COMPONENTS
   roscpp
   rospy
   std_msgs
   sensor_msgs
   geometry_msgs
   message_generation
)

add_message_files(
      FILES
      navdata.msg
      rovstatus.msg
      motortarget.msg
  )

add_service_files(
    FILES
)

generate_messages(
   DEPENDENCIES
      std_msgs
      sensor_msgs
      geometry_msgs
)

catkin_package(
      INCLUDE_DIRS include
      LIBRARIES openrov
```

```
     CATKIN_DEPENDS roscpp rospy message_runtime
                std_msgs sensor_msgs geometry_msgs
     DEPENDS system_lib
)

include_directories(
   ${catkin_INCLUDE_DIRS}
)
```

Además, crearemos un directorio nombrado launch en el que se creará el fichero rosbridge.launch:

4.10: websocket.launch

```
<launch>
   <include file="$(find
      rosbridge_server)/launch/rosbridge_websocket.launch"/>
</launch>
```

Este código conduce a la instalación de rosbridge_server e iniciará la conexión con ROV.

Para finalizar, y antes de ejecutar el .launch, deberemos ejecutar otro fichero, llamado set_remote_robot, el cual está configurado de la siguiente manera:

4.11: set_remote_robot

```
   export ROS_IP=192.168.7.1  #YOUR IP
   export ROS_HOSTNAME=saulibanez.local  #YOUR MACHINE
   export ROS_MASTER_URI=http://saulibanez.local:11311
```

4.3. Conexión de ROS en OpenROV

Habiendo realizado todos los pasos anteriores, ya estamos preparados para realizar la conexión de ROS en OpenROV.

Empezaremos conectando el OpenROV (solo con el cable MiniUSB en el portátil) abriremos un terminal y realizaremos un SSH a el BeagleBone (ssh rov@192.168.7.2).

Una vez conectados, desactivaremos el proxy (sudo /etc/init.d/openrov-proxy stop) y reinicamos elrobot (sudo /etc/init.d/openrov restart).

En otro terminal, ejecutaremos un roscore, que iniciará el ROS Master.

```
roscore http://saulibanez:11311/
saul@saulibanez: ~/catkin_ws    roscore http://saulibanez:11311/
saul@saulibanez:~$ roscore
... logging to /home/saul/.ros/log/14a24092-1b21-11e8-8f89-485ab613c113/roslaunc
h-saulibanez-32411.log
Checking log directory for disk usage. This may take awhile.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://saulibanez:46558/
ros_comm version 1.12.12


SUMMARY
========

PARAMETERS
 * /rosdistro: kinetic
 * /rosversion: 1.12.12

NODES

auto-starting new master
process[master]: started with pid [32424]
ROS_MASTER_URI=http://saulibanez:11311/

setting /run_id to 14a24092-1b21-11e8-8f89-485ab613c113
process[rosout-1]: started with pid [32437]
started core service [/rosout]
```

Figura 4.5: Roscore

Continuaremos lanzando el programa rosbridge.launch, que inciará el nodo de rosbridge para comunicare con el robot.

Ejecutando el código:

4.12: launch

```
$ roslaunch openrov rosbridge.launch
```

```
/opt/ros/kinetic/share/rosbridge_server/launch/rosbridge_websocket.launch http://saulibanez.local:11311
saul@saulibanez: ...   /opt/ros/kinetic/s...   saul@saulibanez: ...   saul@saulibanez: ...   saul@saulibanez: ...
 * /rosdistro: kinetic
 * /rosversion: 1.12.12

NODES
  /
    rosapi (rosapi/rosapi_node)
    rosbridge_websocket (rosbridge_server/rosbridge_websocket)

auto-starting new master
process[master]: started with pid [5842]
ROS_MASTER_URI=http://saulibanez.local:11311

setting /run_id to d67ecea4-0697-11e8-abcb-485ab613c113
process[rosout-1]: started with pid [5855]
started core service [/rosout]
process[rosbridge_websocket-2]: started with pid [5865]
process[rosapi-3]: started with pid [5869]
registered capabilities (classes):
 - rosbridge_library.capabilities.call_service.CallService
 - rosbridge_library.capabilities.advertise.Advertise
 - rosbridge_library.capabilities.publish.Publish
 - rosbridge_library.capabilities.subscribe.Subscribe
 - <class 'rosbridge_library.capabilities.defragmentation.Defragment'>
 - rosbridge_library.capabilities.advertise_service.AdvertiseService
 - rosbridge_library.capabilities.service_response.ServiceResponse
 - rosbridge_library.capabilities.unadvertise_service.UnadvertiseService
[INFO] [1517410985.144924]: Rosbridge WebSocket server started on port 9090
[INFO] [1517413486.040718]: Client connected.  1 clients total.
[INFO] [1517413486.452515]: [Client 0] Subscribed to /openrov/cmd_rate
[INFO] [1517413486.456547]: [Client 0] Subscribed to /openrov/cmd_pose
[INFO] [1517413514.254572]: Client disconnected. 0 clients total.
[INFO] [1517413528.973119]: Client connected.  1 clients total.
[INFO] [1517413529.072633]: [Client 1] Subscribed to /openrov/cmd_rate
[INFO] [1517413529.078050]: [Client 1] Subscribed to /openrov/cmd_pose
```

Figura 4.6: Launch

También se comprobará en el terminal de OpenROV que la conexión se ha realizado

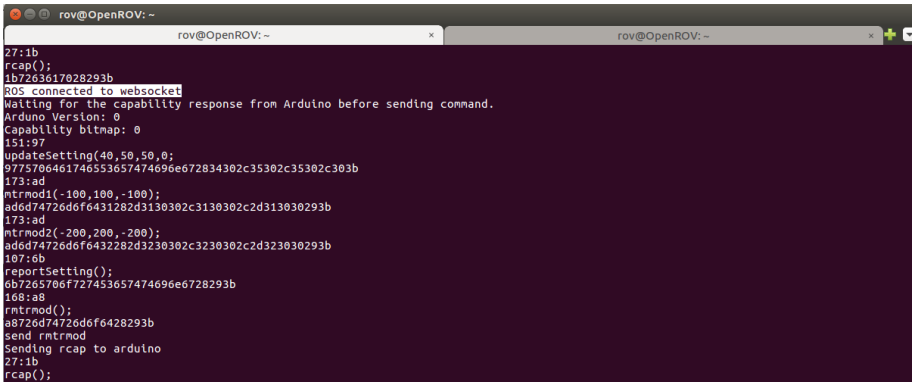


Figura 4.7: Conexión en ROV

Para asegurarse de que todo esté funcionando, abrimos otra pestaña en el terminal e ingresaremos:

4.13: rostopic list

```
$ rostopic list
```

Esto mostrará una lista de los temas que están siendo publicados por el ROV, así como algunos temas predeterminados.

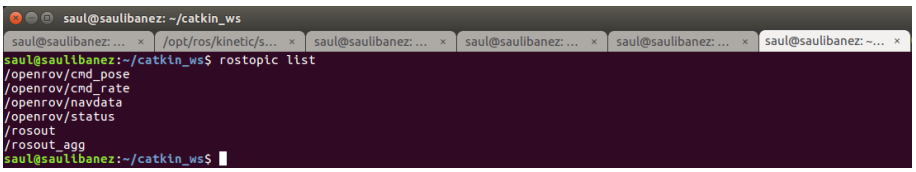


Figura 4.8: rostopic list

Para ver información sobre el nodo en particular podremos ejecutar el comando: rostopic info.

4.14: rostopic info

```
$ rostopic info
```

```
saul@saulibanez: ~/catkin_ws
saul@saulibane... | /opt/ros/kinetic... | saul@saulibane... | saul@saulibane... | saul@saulibane... | saul@saulibanez...
saul@saulibanez:~/catkin_ws$ rostopic info /openrov/cmd_rate
Type: geometry_msgs/Twist

Publishers: None

Subscribers:
 * /rosbridge_websocket (http://saulibanez.local:38123/)


saul@saulibanez:~/catkin_ws$ rostopic info /openrov/navdata /
Usage: rostopic info /topic

rostopic: error: you may only specify one topic name
saul@saulibanez:~/catkin_ws$ rostopic info /openrov/navdata
Type: openrov/navdata

Publishers:
 * /rosbridge_websocket (http://saulibanez.local:38123/)

Subscribers: None


saul@saulibanez:~/catkin_ws$ rosm
rosmake       rosmaster     rosmsg        rosmsg-proto
saul@saulibanez:~/catkin_ws$ rostopic info /openrov/navdata
Type: openrov/navdata

Publishers:
 * /rosbridge_websocket (http://saulibanez.local:38123/)

Subscribers: None


saul@saulibanez:~/catkin_ws$ rosmsg show /openrov/navdata
Unable to load msg [/openrov/navdata]: invalid name [/openrov/navdata]
```

Figura 4.9: rostopic info (1)

```
saul@saulibanez: ~/catkin_ws
saul@saulibane... | /opt/ros/kinetic... | saul@saulibane... | saul@saulibane... | saul@saulibane... | saul@saulibanez...
saul@saulibanez:~/catkin_ws$ rostopic info /openrov/cmd_pose
Type: geometry_msgs/Pose

Publishers: None

Subscribers:
 * /rosbridge_websocket (http://saulibanez.local:38123/)


saul@saulibanez:~/catkin_ws$ rostopic info /openrov/cmd_rate
Type: geometry_msgs/Twist

Publishers: None

Subscribers:
 * /rosbridge_websocket (http://saulibanez.local:38123/)


saul@saulibanez:~/catkin_ws$ rostopic info /openrov/navdata /
Usage: rostopic info /topic

rostopic: error: you may only specify one topic name
saul@saulibanez:~/catkin_ws$ rostopic info /openrov/navdata
Type: openrov/navdata

Publishers:
 * /rosbridge_websocket (http://saulibanez.local:38123/)

Subscribers: None


saul@saulibanez:~/catkin_ws$ rosm
rosmake       rosmaster     rosmsg        rosmsg-proto
saul@saulibanez:~/catkin_ws$ rostopic info /openrov/navdata
Type: openrov/navdata
```

Figura 4.10: rostopic info (2)

Capítulo 5

Pruebas de Campo

En este capitulo, se adjuntarán unos vídeos comprobando el funcionamiento del robot.

1. En el primer vídeo, después de haber realizado la soldadura de los motores y comprobado que existe conectividad, se verifica que su funcionamiento y rotación es la adecuada.

   `https://www.youtube.com/watch?v=pck8Bpla-5w`

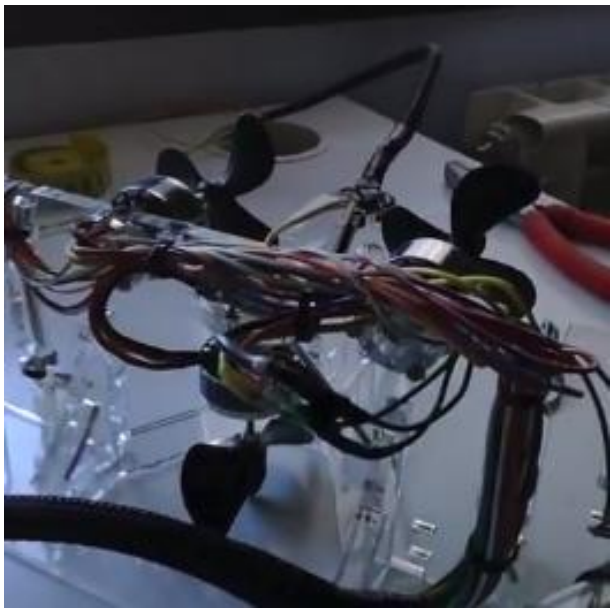


Figura 5.1: Prueba de motores

2. Una vez montado el robot, antes de llevarlo al exterior, se comprueba que es estanco y que el hardware no sufre ningún daño, para este proceso, llenamos la bañera de agua e introducimos el robot.

   También se comprueba que funcionan todas las aplicaciones instaladas y que puede permanecer más de 6 horas bajo el agua sin incidentes.

   `https://www.youtube.com/watch?v=jqZOBc_-mvU`

Figura 5.2: Prueba de funcionamiento en la bañera

3. Una vez comprobados los dos apatados anteriores, es momento de comprobar su funcionamiento en un entorno más hostil, para ello, nos hemos dirigido a un lago.

   Antes de sumergirlo, comprobaremos que todo funciona correctamente y que no se necesita internet para interactuar con el robot.

https://www.youtube.com/watch?v=0LhODkpdkcw

Figura 5.3: Prueba de funcionamiento en el lago

4. En este video, se comprueba que el robot puede navegar por el lago sin ningún problema.

https://www.youtube.com/watch?v=KLFG9klwu_Q

Figura 5.4: Prueba de navegación en el lago

5. Se realiza una prueba de emersión para comprobar que el robot puede volver a la superficie sin problemas.

`https://www.youtube.com/watch?v=Q6qG2w9rGW0`

Figura 5.5: Prueba de emersión del lago

6. En el último video, vemos gracias a la web, todo lo que el robot está visualizando en el fondo del lago.

`https://www.youtube.com/watch?v=svbzpc0WSwE`

Figura 5.6: Prueba de visualización del lago en la web

Capítulo 6

Conclusiones

Los objetivos de este trabajo de fin de grado han sido dos: por un lado, presentar a los robots submarinos, en particular, OpenROV y por otro lado, la integración de ROS, el framework de robótica más extendido y utilizado con la tecnología del ROV.

Se realizó el montaje y la configuración de cero, por lo que he podido aprender sobre la tecnología del ROV, además de explorar los paquetes necesarios de ROS para realizar la comunicacion entre ROS y OpenROV.

Tras finalizar el montaje y el sistema del OpenROV aprendí a solventar imprevistos que no se contemplaban tanto de hardware como de software. En este aspecto, la comunidad de OpenROV ha sido de inestimable ayuda en algunos momentos.

Se comprobó que el robot es estanco y puede navegar bajo la superficie marina, que todos los controles mencionados en los puntos anteriores funcionan correctamente (láser, cámara, luces, etc) y que es probable que aparezcan algunos de los problemas comunes (por ejemplo, el vaho).

Como OpenROV tiene software libre, se pudo instalar los paquetes necesarios para realizar la comunicación con ROS y se comprobó que existía la comunicación entre las dos plataformas. Al final, el OpenROV puede mejorar bastante si se siguen implementando nodos de ROS para su utlilización (por ejemplo, la cámara, usando el paquete sensors, la velocidad, utilizando geometry_msgs, etc).

Para finalizar, otro de los siguientes pasos que se puede realizar con este TFG, es intentar comprobar si la leyenda de Nessie es cierta.

Bibliografía

[1] B. Adams, "Ros integration with openrov," http://blog.openrov.com/ros-integration-with-openrov/, 2015.

[2] Anónimo, "Una introducción a los robots móviles," http://www.aadeca.org/pdf/CP_monografias/monografia_robot_movil.pdf, 2008.

[3] W. Garage, "Ros," http://www.ros.org/, 2010.

[4] W. W. Royce, "Modelo en cascada," https://es.wikipedia.org/wiki/Desarrollo_en_cascada, 1970.

[5] E. Stackpole and D. Lang, "Openrov," https://openrov.dozuki.com/ , and https://openrov.com, 2015.

[6] D. D. Tarascó, "Desarrollo de sistema de telemetría para teleoperación de robot submarino," Proyecto Fin de Carrera, Universidad Simón Bolívar, 2016.